PROGRAMACIÓN DE APPS – ANDROID (NATIVO) ENTORNO DE DESARROLLO

Para implementar una aplicación nativa, vamos a utilizar el entorno de desarrollo Eclipse.

Ilustración 1 - Pantalla de Carga del Entorno de Desarrollo

Una vez creada y compilada nuestra primera aplicación en android, se puede observar que se han creado multitud de carpetas y ficheros dentro del proyecto. Antes de empezar a programar la aplicación vamos a hacer un repaso por el significado de las carpetas que nos aparecen asociadas al proyecto, y que se han generado de manera automática.

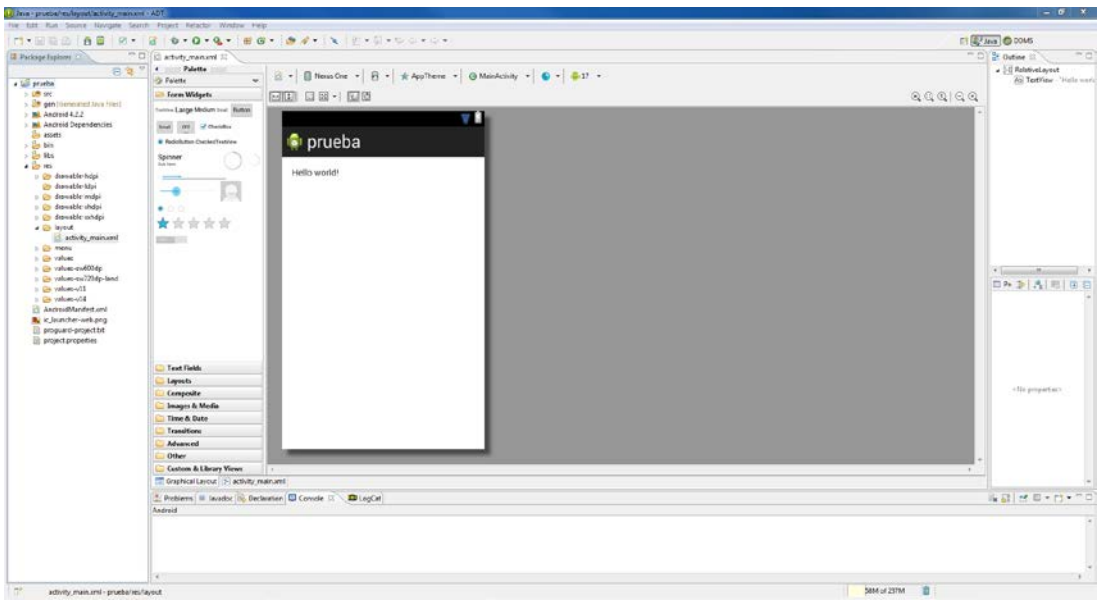

Ilustración 2 - Pantalla de creación de la primera aplicación

ESTRUCTURA DE UNA APLICACIÓN

Ahora podemos ver en la pestaña **Package Explorer** la lista de todos archivos y directorios que se han generado. Vamos a ver el significado de las carpetas fundamentales.

/src: en este directorio es donde se almacenarán **nuestros archivos de código fuente**. Por defecto se incluye un tipo de actividad que se ejecuta cuando la aplicación se abre.

/gen: aquí aparecerán archivos que genera el entorno de programación de forma automática, como por ejemplo el archivo R.java o el BuildConfig.java. Estos archivos **no deberemos editarlos**, ya que será Eclipse el que se encargará de modificarlos automáticamente según otros parámetros del proyecto.

/Android 4.2.2: es la librería de desarrollo de Android (SDK). Podemos ir desplegando el árbol para ver los paquetes y clases que incluye el SDK. Cuando añadamos nuevas librerías o paquetes a nuestro proyecto, aparecerán de la misma forma.

/assets: es un directorio para guardar recursos que utilice tu aplicación. Para acceder a los recursos de este directorio necesitaremos usar la clase AssetManager para leer los datos como un stream de bytes, por lo que es preferible utilizar el directorio **/res** ya que el acceso a los archivos es mucho más sencillo.

/res: es el directorio principal de recursos (resources). Aquí guardaremos imágenes o archivos multimedia que utilice nuestra aplicación. Los recursos colocados en este directorio son fácilmente accesibles desde la clase R.

/res/drawable-?dpi: es el directorio de recursos gráficos o imágenes que utilizará nuestra aplicación. Tiene diversos subdirectorios: drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xhdpi y drawable-xxhdpi,, en el que guardaremos las imágenes **dependiendo de la densidad de puntos por pulgada** que tenga el dispositivo en el que se ejecute la aplicación. Para realizar un cálculo del tamaño que tienen que tener las imágenes en android, atendiendo a la densidad de pixeles de la pantalla, debemos tomar por referencia las imágenes para mdpi, y aplicar la siguiente escala para el resto de las imágenes:

- **Xhdpi = 2.0 (200x200)**
- **Hdpi = 1,5 (150x150)**
- **Mdpi = 1 (tomado como base) (100x100)**
- **Ldpi = 0,75 (75x75)**

Las imágenes generadas siguiendo esta escala, tendrán el mismo nombre, pero se colocarán en la carpeta correspondiente a cada densidad. Para obtener información más detallada sobre esto se puede visitar la página oficial de desarrollador de Android, y las estrategias de diseño del apartado de diseño de una aplicación del mismo manual.

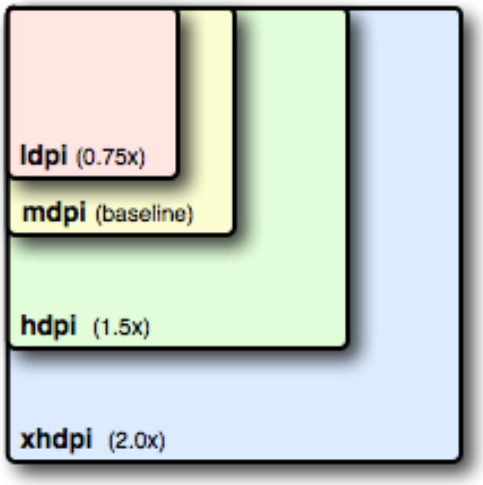

Ilustración 3 - Escala de las imágenes

/res/layout: Al utilizar programación MVC (Modelo Vista Controlador), debemos separar el código de la aplicación de la interfaz gráfica. En este directorio es donde crearemos los archivos xml que definen las vistas que serán utilizadas en la aplicación. Desde código indicaremos que archivo xml queremos que use en cada momento. Gracias a esto, se nos facilita mucho modificar el código, ya que no tendremos que modificar el entorno gráfico, y viceversa. Para la visualización en diferentes tamaños de se pueden crear las carpetas siguientes:

- res/layout/mi_layout.xml = vista para pantallas por defecto
- res/layout-small/mi_layout.xml = vista para pantallas pequeñas
- res/layout-large/mi_layout.xml = vista para pantallas grandes

- res/layout-xlarge/mi_layout.xml = vista para pantallas extra grandes
- res/layout-xlarge-land/mi_layout.xml = vista para pantallas extra grandes (modo apaisado)
- res/layout-xlarge-port/mi_layout.xml = vista para pantallas extra grandes (modo retrato)

Al igual que los diferentes dispositivos difieren en densidad de pixeles por pulgadas, también existe una variedad muy amplia en cuanto a **tamaños de pantalla y su orientación (portrait = modo retrato y landscape = modo apaisado)**. Como podemos apreciar en el tipo de carpetas que podemos crear, android separa los tamaños en cuatro grandes grupos. Para una información detallada puede visitar el apartado de métricas del manual oficial.

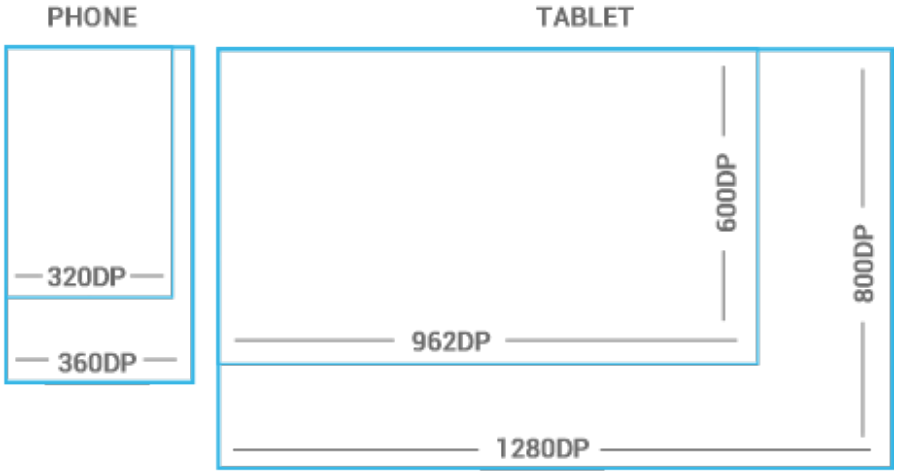

Ilustración 4 - Tamaños de pantallas

De manera global se puede visitar con total detalle el apartado de soporte para múltiples pantallas.

/res/menu: En esta carpeta se definirán las características de nuestros menús. Android suministra la posibilidad de crear diferentes tipos de menús atendiendo a las necesidades que cada aplicación requiera.

Recordamos que si estamos programando para Android 2.3 o inferior, el contenido de su menú será visible cuando el usuario presione en el menú de opciones.

Ilustración 5 - Menu para Android 2.3 o inferior

En el caso de la programación para Android 3.0 o superior, los ítems del menú estarán situados en la barra de acción.

Ilustración 6 - Menú para Android 3.0 o superior

/res/values: Debemos separar las cadenas de texto o arrays, los estilos y las dimensiones que usará la interfaz del código. Este apartado cobra importancia en la localización de nuestra aplicación. Por ejemplo, podremos definir un archivo strings.xml en **/res/values-es/** y otro en **/res/values/** El primero se utilizará automáticamente si el usuario tiene configurado el terminal en el idioma español, mientras que en el resto de casos se utilizará el archivo que se encuentra en /res/values/.

En el siguiente enlace encontramos el código ISO 639-1 de 136 países.

AndroidManifest.xml: es el archivo de configuración global de nuestra aplicación. En este fichero informaremos al sistema de las actividades o servicios que ejecutará nuestra

aplicación y los permisos especiales que necesita utilizar en el caso de que quiera acceder a recursos compartidos del sistema, como el acceso a la lista de contactos o uso del GPS. En resumen, este fichero describe las características fundamentales de la aplicación. Uno de los elementos fundamentales es el <uses-sdk> que define la compatibilidad con las diferentes versiones de android. Para más información podemos leer el apartado dedicado a esto en el manual oficial de desarrollo.

default.properties: es un fichero que se genera automáticamente y del que no deberemos preocuparnos inicialmente.

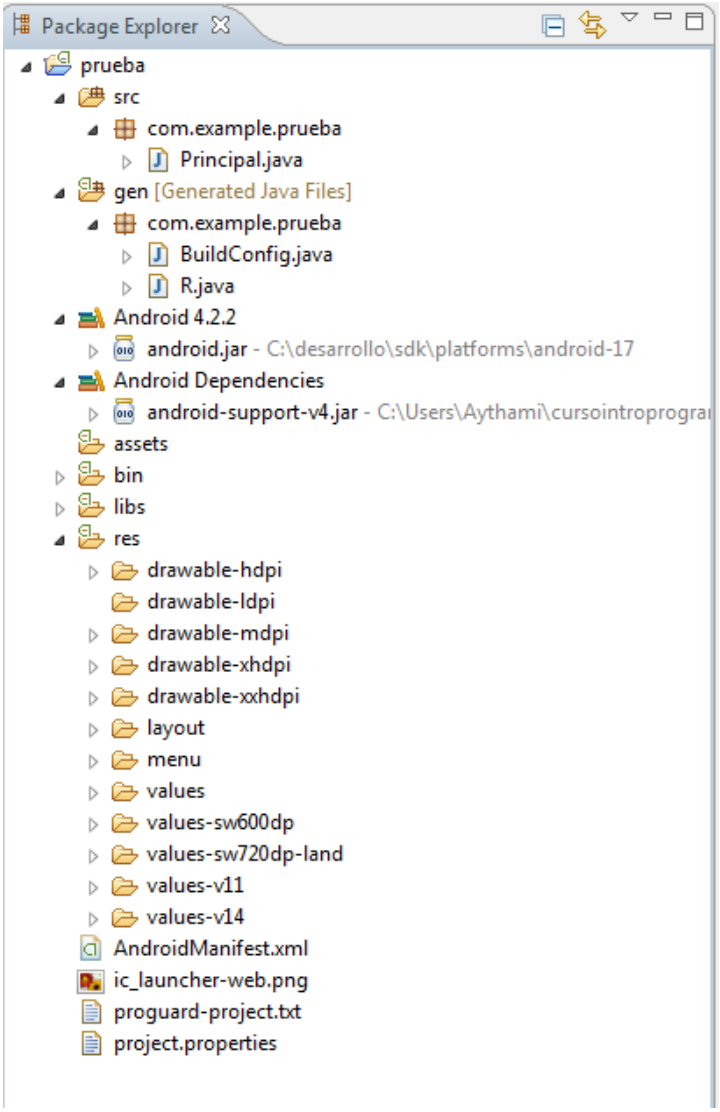

Ilustración 7 - Vista general ejemplo del Package Explorer

ANEXO – RECURSOS ALTERNATIVOS

Para cualquier tipo de recurso, puedes especificar cuáles van a ser los recursos usados por defecto y los recursos alternativos. Los **recursos usados por defecto** son aquellos que deberían ser usados sin considerar la configurar específica del dispositivo o cuando no hay recursos alternativos que coincidan con la configuración actual.

Los **recursos alternativos** son aquellos que hemos diseñado para que sean usados con una configuración específica. Para especificar que un grupo de recursos están asociados a una configuración específica, debemos **añadir un sufijo al nombre de la carpeta** relacionándola con una determinada configuración. Se debe seguir el siguiente formato para nombrar la carpeta:

res/tipoRecurso-cualificador

A continuación puedes encontrar los posibles nombres de cualificadores que puedes usar, agrupados por categorías:

MCC y MNC:

Código de país de móvil (mobile country code MCC) seguido opcionalmente del código de red de móvil (mobile network code) de la tarjeta SIM que tenga el dispositivo.

Por ejemplo, mcc214-mnc07 para Movistar, mcc214-mnc01 para Vodafone o mcc214-mnc03 para Orange.

Idioma y región:

El código de idioma está definido por 2 letras según ISO 639-1, y puede ir seguido opcionalmente del código de región ISO 3166-1-alpha-2precedido del carácter 'r'.

Por ejemplo, "es" para Español de cualquier región, o "es-rES" para español de España y "es-rMX" para español de México. "en" es inglés para cualquier región, "en-rUS" es inglés de Estados Unidos y "en-rGB" es inglés de Gran Bretaña.

Menor tamaño de pantalla:

Indica el tamaño más pequeño de la pantalla (considerando tanto alto como ancho). Esta medida es indiferente de la orientación de la pantalla. Se indica la medida precedida de "sw" y seguida de "dp", por ejemplo sw320dp o sw720dp. **Desde API 13.**

Ancho disponible:

Especifica el ancho mínimo de pantalla disponible, cambiando su valor cuando la orientación de la pantalla varíe entre apaisado o vertical considerando en cada caso el ancho correspondiente. Se indica la medida precedida de "w" y seguida de "dp", por ejemplo w720dp o w1024dp. **Desde API 13.**

Altura disponible:

Especifica la altura mínima de pantalla disponible de manera similar a la anterior. Por ejemplo: h720dp o h1024dp. **Desde API 13.**

Tamaño de pantalla:

small: 320x426 dp aprox.

normal: 320x470 dp aprox.

large: 480x640 dp aprox.

xlarge: 720x960 dp aprox. **Desde API 9.**

Aspecto de la pantalla:

long: Pantallas largas como WQVGA, WVGA y FWVGA.

notlong: Pantallas menos largas como QVGA, HVGA y VGA.

Orientación de la pantalla:

port: (Portrait) Modo retrato (vertical)

land: (Landscape) Modo apaisado (horizontal)

Modo de interfaz de usuario(desde API 8):

car: Dispositivo conectado a un soporte (dock) de coche.

desk: Dispositivo conectado a un soporte de escritorio.

television: Dispositivo mostrado en una televisión. **Desde API 13.**

appliance: Dispositivo actuando como appliance (diseñado para actuar para un propósito específico), sin pantalla.

Modo nocturno (**desde API 8**):

night: En horario nocturno.

notnight: En horario diurno.

Densidad de puntos de pantalla(dpi):

ldpi: 120dpi aprox.

mdpi: 160dpi aprox.

hdpi: 240dpi aprox.

xhdpi: 320dpi aprox. **Desde API 8**

nodpi: Se puede usar para imágenes que no se desea escalar en función de la densidad de pantalla.

tvdpi: Entre mdpi y hdpi. 213dpi aprox. **Desde API 13**

Hay un ratio de escala de 3:4:6:8 entre los cuatro primeros tipos de densidad. Así, una imagen de 9x9 en ldpi es de 12x12 en mdpi, 18x18 en hdpi y 24x24 en xhdpi.

Tipo de pantalla táctil:

notouch: No tiene pantalla táctil.

finger: Tiene pantalla táctil.

Disponibilidad de teclado:

keysexposed: El dispositivo tiene un teclado físico disponible.

keyshidden: Tiene un teclado físico pero está oculto.

keyssoft: Tiene un teclado virtual (por software).

Tipo de teclado físico:

nokeys: El dispositivo no tiene teclas físicas.

qwerty: Tiene un teclado físico tipo qwerty.

12key: Tiene un teclado físico de 12 teclas.

Disponibilidad de teclas de navegación:

navexposed: Las teclas de navegación están disponibles.

navhidden: Las teclas de navegación no están disponibles.

Método principal de navegación no táctil:

nonav: El dispositivo no tiene otro medio de navegación que no sea la pantalla táctil.

dpad: Tiene un pad (teclado) direccional.

trackball: Tiene un trackball (bola).

wheel: Tiene ruedas para la navegación (poco común).

Los nombres de cualificadores deben seguir además una reglas:

Se pueden especificar varios cualificadores a un mismo conjunto de recursos, separándolos por guiones. Por ejemplo: drawable-en-rUS-land se aplicaría a dispositivos configurados con el idioma inglés de Estados Unidos y cuando esté en modo apaisado.

Los cualificadores deben estar en el orden de la lista anterior. Por ejemplo:

Incorrecto: drawable-hdpi-port

Correcto: drawable-port-hpdi

Las carpetas de recursos alternativos no pueden ser anidadas. Por ejemplo:

Incorrecto: res/drawable/drawable-en

Correcto: res/drawable-en

Los nombres de cualificadores **no consideran las diferencias entre mayúsculas y minúsculas**.

Sólo es válido un valor para cada tipo de cualificador. Por ejemplo:

Incorrecto: drawable-es-fr

Correcto: drawable-es y otra carpeta drawable-fr.

Android utilizará automáticamente en la aplicación los recursos basados en la configuración actual del dispositivo. Cada vez que se solicita un recurso, Android comprueba si hay carpetas de recursos alternativos que contengan el recurso solicitado y entonces localiza el recurso que mejor se adapta a la configuración actual. Si no hay recursos alternativos para la configuración actual, Android **usará los recursos correspondientes por defecto** (los que no incluyen un cualificador).